Evolution and use of generic markup languages

Pamela L. Gennusa

1 Introduction

This article will provide:

- a brief overview of the key concepts of generic markup languages used for structured documents;
- an introduction to the two most prominent languages in use, SGML and XML, and their respective families of related standards; and
- reflections on the impact of the use of these languages to structure documents in the real world.

I hope this article will provide an understanding of why generic markup languages were created, why their details are so energetically debated, how they will contribute to the next generation of document processing technologies, and how their use can help companies better realize business goals.

1.1 The challenge for publishers

In the beginning

Corporate and commercial publishers have long been held captive by publishing software and hardware vendors. If a publisher had thousands of pages encoded in a single vendor's proprietary markup language, it was very time consuming and costly for the publisher to move to the use of another vendor's software or to re-purpose the

information. The markup embedded in the documents was typically format oriented, describing how the text should be laid out on the page. Each typesetting vendor used a different set of commands to achieve the same result, thus making interchange of documents between typesetting systems very difficult. When the only output was printed documents, this situation was minimally tolerable. However, when publishers wanted to produce their documents on multiple media, the publishers' tolerance level quickly fell. The use of an internationally agreed generic markup language meant that publishers could create information once and automate its reuse multiple ways, saving time and effort.

1.2 Key concepts of generic markup languages

content vs. formatting markup

The use of a single, internationally agreed syntax was not the only reason why a generic markup language was the appropriate solution for publishers that wanted to publish to multiple media. A fundamental concept in the use of generic markup languages is a move from formatting markup to content markup.

For example, in early typesetting systems, a date at the top of a letter might be encoded as a 10 point type on 12 points of leading, using the Times Roman font, and set with a 20 pica indent quad left on the line. Later systems allowed the use of macros, so the above set of micro formatting instructions might be referenced as a whole by a macro named `indent-head`. The same macro might be used for the signature line in the letter as well. Using a generic markup language, the date would be identified as a *date*. Additionally, the components of the date might be further identified as *day*, *month*, and *year*. The signature line might be identified as `signature-block` and further decomposed into *signer* and *position*.

One advantage of the use of content markup is the ability to delimit components of documents into named structures and, thus, to be able to find and use specific pieces of content. In the example above, a query could be given to a database to find all letters written after 6 August 1997 and signed by the Comptroller. Without content markup, letters that reference a later date or the Comptroller in the body of the document would also satisfy the search.

2 The rise of generic markup languages

2.1 GenCode®, GML, and SGML

As long ago as 1967 (for the source of this information, see *A brief History of the Development of SGML*, SGML Users' Group 1989), Mr. William Tunnicliffe, chairman of the *Graphic Communications Association (GCA)* Composition Committee, made a presentation to a meeting of the *Canadian Government Printing Office* on the topic of the separation of content from format. Coincidentally, during the late 1960s, Stanley Rice, a book designer, proposed the idea of a universal catalogue of parameterized 'editorial structure' tags.

Generic Coding project

Understanding the significance of these proposals, the GCA established a generic coding project in their Composition Committee. The committee, recognizing that different generic codes were needed for different kinds of documents and that smaller documents could be incorporated as elements of larger ones, developed the 'GenCode concept' and registered the *GenCode* trademark. This project later evolved into the GenCode Committee.

Generalized Markup Language (GML)

In 1969, Charles Goldfarb, together with Edward Mosher and Raymond Lorie, invented the *Generalized Markup Language (GML)* at IBM. GML was created to allow text editing, formatting, and information retrieval systems to share documents. Following the initial development of GML, Goldfarb continued his research into document structures and defined additional concepts in generic markup.

ANSI project for a text description language

In 1978, Goldfarb was asked to join the *American National Standards Institute (ANSI)* Computer Languages for the Processing of Text committee and to lead a project for a text description language standard based on GML. The GCA's GenCode committee members became the foundation of the membership of this ANSI committee. The first working draft was published in 1980. By 1983, the GCA was able to publish the sixth working draft as an industry standard (GCA 101-1983), which was adopted by the *Internal Revenue Service (IRS)* and the *US Department of Defense (DOD)*.

publication of ISO 8879

The project was also authorized by the *International Organization for Standardization (ISO)*. In early 1985, the draft proposal for an ISO standard was published and, by October 1985, the draft international standard was ready for balloting. Following a year of review, comment resolution, and production, *ISO 8879 Standard Generalized Markup Language (SGML)* was published on 15 October 1986,

using an SGML system at the *European Particle Physics Laboratory (CERN)*.

SGML is a meta-language

So, how can we define SGML? Primarily, SGML is a meta-language, that is, a language for creating markup languages. It has an internationally recognized syntax that is human-readable and machine processible. The syntax is used to define document classes and to mark up individual instances of a class. Although SGML didn't become an international standard until 1986, from the preceding history, it can be seen that the need for SGML and the concept of generic markup had been apparent for a number of years. By the time SGML became an ISO standard, there were applications running in conformance with the Draft International Standard and even earlier versions because this was a standard that many industries were anxious to use.

2.2 HyTime

The advent of electronic delivery of documents opened the door on the use of highly internally and externally referenced content. The market need grew quickly and a number of vendors created proprietary products to address that need. Again, the user was faced with using interesting technology, but being tied to a single vendor.

publication of ISO 10744

The SGML standard provided only the most basic support for the embedding of cross-references. In response to the burgeoning need in the hypertext marketplace, ISO created *ISO 10744 Hypermedia/Time-based Structuring Language (HyTime)* in 1992 to provide a standard syntax for the description of such linkages. The second edition of HyTime was published in 1997.

2.3 DSSSL

Because SGML is typically only used to describe the content and not the format of documents, users were also interested in a way to interchange vendor-independent style specifications. ISO responded with the creation of *ISO 10179 Document Style Semantics and Specification Language (DSSSL)* in 1996. Using a DSSSL specification, it is possible to specify the intent of formatting for each element-in-context within an SGML document as well as the general pagination rules.

Work on DSSSL began in the mid-1980s. In the early 1990s, it was balloted and approved as an International Standard. However, the ISO Working Group was not satisfied that the standard had met its goals. They continued work on the standard until it was again balloted in late 1995. DSSSL is considered a sound and comprehensive standard, albeit a complex one.

publication of ISO 10179

2.4 CALS

In 1983, the *US Department of Defense* adopted the use of the GCA's industry standard built on the sixth working draft of SGML. However, this was just a placeholder until SGML was published as an ISO standard. In September 1985, a full year before SGML achieved the status of an ISO standard, it was cited for use in a major US DOD initiative regarding electronic data. CALS (originally for Computer Aided Logistic Support, see References) endorsed the use of standards for the encoding of text, graphics, product data, logistic information, etc. The principal standards initially recommended for use were *ISO 8879 Standard Generalized Markup Language (SGML)*, *ISO 8632 Computer Graphics Metafile (CGM)*, and *ASME/ANSI Y14.26M Initial Graphics Exchange Specification (IGES)*.

the 800-pound (363.63-kg) gorilla

The adoption of SGML by the DOD was probably one of the greatest contributions to its growth. The US DOD is often referred to as the 800-pound gorilla. The joke goes: 'What do you give an 800-pound gorilla? ... Whatever it wants.' The DOD's decision to make SGML part of its CALS initiative gave rise to a generation of vendors who believed that there would be a market if they were to develop new products or redesign their current products to use SGML.

2.5 OS/FOSIs

Due to the protracted development period of DSSSL (approximately 10 years), the CALS project office was not prepared to wait until DSSSL was ready to be able to non-proprietarily specify the intent of formatting. Therefore, the CALS project office approved the development of an interim style specification called an *Output Specification (OS)*. The OS defined the (DOD's) universe of technical documentation formatting parameters available for application to any DTD. A Formatting Output Specification Instance (FOSI) was an instance of the OS, applied to a specific DTD. A small number of

an interim style specification

commercial products implemented the FOSI in editorial and paper output environments. It was a step in the right direction, but the OS/FOSI suffered from the early rush in its development. There were certain formatting characteristics that could not be expressed in the language. Thus, for the sake of expediency, the OS/FOSI standard permitted some of the formatting specification to be provided in a non-standardized way. Though well intentioned, such trap doors left the user again with a non-portable formatting specification.

2.6 Interactive Electronic Technical Manuals

an electronic delivery model

The US DOD was also working on a number of electronic delivery models. Proposals for a DOD-wide standard were created in both the Air Force and the Navy. A tri-service committee ultimately developed a set of three standards defining a database structure for the underlying content, a formatting style for the user interface, and a quality assurance specification. This type of delivery is known as an *Interactive Electronic Technical Manual (IETM)*. The IETM specifications broke new ground by incorporating a holistic view of a set of documentation, using HyTime constructs to model relationships, and by modeling the interactions that might be used with the documentation. For example, a procedure says to test a voltage and then, based on the voltage, cross-references other places in the information set. This interaction might take place between the documentation and the maintenance engineer. Alternatively, it might take place between the documentation and a piece of automated test equipment. IETMs have been adapted for use across nations. For example, in the UK, *Def Stan 00-60* cites the use of *Interactive Electronic Technical Publications (IETPs)*, referencing a UK version of the US IETM specifications.

3 Advent of the Web

But that was yesterday

In the early 1990s, the Internet was gaining users. To make the Internet easier to access, the concept of the World-Wide Web was born. The WWW project, started by Tim Berners-Lee while at CERN, seeks to build a 'distributed hypermedia system'.

Early on, the *HyperText Markup Language (HTML)* was developed for use on the Web. The first version of HTML was not strictly SGML, although it tried to 'look' like SGML through the use of tags.

Due to the efforts of a number of SGML proponents, most notably, Yuri Rubinsky, HTML was 'SGMLized' by the time HTML 2.0 came out. The development of the HTML DTD made it possible to programmatically ensure that HTML documents did in fact conform to a common definition.

However, most browsers accept less than perfect HTML, or perhaps it is more accurate to say, they accept more than a widely accepted definition of HTML. Each browser vendor developed proprietary extensions, allowing 'tags' that were not in the then current definition of HTML.

3.1 Pressure for less than SGML, more than HTML

Although SGML has been a stable standard since 1986, it was created at a time when information technology for text processing was in a state of immaturity. For example, at the beginning of the development of SGML, there were not even word processors that used programmable keystrokes, nor was the concept of SGML editing software established (i.e., that software would enter all of the markup commands). During that period, standard working practice was to have text keyboarded by a typesetting company or typing pool that charged by the keystroke. Thus, parts of the SGML standard were written to address the ability to 'minimize' the markup through various conventions.

complex features made the SGML standard difficult

The use of minimization and other complex features made the SGML standard difficult and time-consuming to implement. With hindsight, and knowledge of today's available software support, it is acknowledged that many of the more complex parts of SGML may not be needed by all users. Further, a more streamlined version of the standard would mean that more software companies could create products for a lower start-up cost, thus increasing the number of more affordable text processing software.

HTML is an application of SGML

The advent of the Web and its global popularity also had an impact. The markup language of the Web, *HyperText Markup Language (HTML)* is an application of SGML. It is a single tag set based on the HTML Document Type Definition. Although very useful for the presentation of a wide variety of document classes, it suffers from the use of formatting over content markup. Even if a corporation's text databases were encoded in SGML, they had to be transformed into HTML, losing the richness and processibility of the content markup in the process. This has meant that client-side processing has been

constrained due to the 'sameness' of information once it is encoded in HTML and that most processing has to take place on the server site, resulting in poor performance and long wait times. As noted above, an alarming and negative consequence was the proliferation of proprietary extensions to the HTML tagging set by individual browser vendors, flinging users of the Web back a decade or two to an age when documents were 'stuck' within a vendor's product.

3.2 The role of the World-Wide Web Consortium (W3C)

request for a 'slimmer' version of SGML

The above conditions led to an initiative by a number of SGML experts to create a 'slimmer' version of SGML. Many of these SGML experts were also involved with the World-Wide Web Consortium (W3C). The strong impetus to cut down SGML for use on the Web became a good reason to house this work within a W3C committee. Led by Jon Bosak and others, the W3C approved the creation of a Working Group to develop an extensible language for use on the Web. The SGML background of those involved meant that they would produce a streamlined version of SGML to achieve this goal. The result of an 18-month effort was a highly successful, well-received, cut-down version of SGML called the *Extensible Markup Language* or *XML*.

3.3 Enter XML

XML = leaner, meaner SGML

XML has all of the characteristics of SGML that made it a strong choice for returning business benefits, such as the use of an internationally agreed syntax; the ability to create well-defined and well-structured documents; and the use of generic, content-based markup. XML has been called both a 'kinder, gentler SGML' and a 'leaner, meaner SGML'. Both are true. For someone just moving to generic markup languages, it is easier to learn and to use. For developers of software and systems, it gets the job done with reduced complexity and lower risk.

Current users of SGML are in an advantageous position in that the majority of their SGML documents may already be very close to XML conformant. A complete transition to XML is far less time-

consuming and costly than the initial adoption of a generic, content-based markup language.

3.4 ... and XLink

XML linking language

Following immediately on to the start of the creation of XML, the Working Group determined that they should also create an *Extensible Linking Language (XLink)*. XLink, originally known as XML-link and then as XLL, is an encompassing term for XML hyperlinking (linking and addressing). It has two major components: XLink (proper) and XPointer. XLink is more robust than the unidirectional cross-referencing in SGML and less complex than the range of facilities in HyTime. XLink (proper), released as a working draft (March 1998), 'specifies constructs that may be inserted into XML resources to describe links between objects'. (A link in this context may be unidirectional as well as multi-ended and typed.)

XPointer, also released as a working draft (March 1998), defines 'constructs that support addressing into the internal structures of XML documents. In particular, it provides for specific reference to elements, character strings, and other parts of XML documents, whether or not they bear an explicit ID attribute.'

3.5 ... and XSL

XML style language

DSSSL is a large and comprehensive specification that took a number of years to complete. It is generally acknowledged to be too much for the Web at some levels (complex pagination) and not enough for the Web (interface behavior) at others. HTML, being a fixed set of tags, was relatively easy to format without further information. The HTML tags had associated formatting characteristics that could be changed on a per instance basis. The use of XML on the Web introduces the concept of markup tags that are unknown to the rendering device. In order for XML to be accepted on the Web, it was necessary to address the issue of how a document creator could specify the formatting to be applied to a set of tags and a browser could interpret that formatting specification.

To address the need to specify formatting for an XML application, the W3C approved another work item called *Extensible Style Language* or *XSL*. The XSL Working Group released a requirements document in May 1998. Even as the specification is being drafted,

vendors are experimenting with and prototyping the output of the Working Group.

3.6 ISO's Web SGML

SGML for Web applications

The creators of XML were all experienced in the use of SGML. They understood the role that an ISO standard plays and the importance of compliance with SGML. Specifically, they knew that the existing SGML community would not be able to easily transition between SGML and XML if the two were not compatible ... and they were all users of SGML. They knew also, that dedicated SGML users wanted to be able to use SGML on the Web and wanted a wider choice of products. Thus, compatibility with SGML became an important design criterion. To ensure that end, communications between the W3C XML Working Group and the ISO SGML Working Group were frequent and extensive.

concept of well-formedness and other differences

Perhaps the most significant difference between XML and traditional SGML was the optionality of the *DOCTYPE Declaration*. Whereas, an SGML document is not complete without it, an XML document may or may not have one. XML brought forth the concept of *well-formedness*. A well-formed document is one in which the structure can be completely determined from the instance. To make this possible, the XML Working Group determined that there would be no tag omission, except in the case of empty elements, where the end tag would be omitted. However, without a DTD, it was not possible to know that an element was empty unless the start tag 'looked' different from other start tags. Thus, it was necessary to introduce the special construct known as the *Empty-Element Tag* (`<tag_name/>`).

SGML/XML compatability

A few other differences also became necessary, such as the use of predefined character entities, the ability to use only a specific subset of the `SHORTTAG` feature and absolutely disallow other parts, and the ability to have an optional or partial DTD. To ensure that 'what was XML, was SGML', the ISO SGML Working Group set about creating two annexes (K [normative] and L [informative]) to the SGML standard. These two annexes, extended SGML to encompass those aspects of XML that were initially outside the scope of SGML.

Today, with the addition of these annexes, XML truly is a subset of SGML, as per its design objectives. Further, SGML processors, if they support Annex K, can be said to support XML and SGML.

3.7 A thought about SGML Mark II

call for a streamlined SGML

In his closing keynote speech at SGML Europe '96, Tim Bray, *Textuality, Inc.*, called for a streamlined version of SGML, one that would be easier to implement and understand. Shortly thereafter, the XML Working Group was formed within the W3C.

new generations of computing systems and the Web technology allow streamlining

However, sometimes I wonder what SGML would have been like if it had been created now, with a dozen years experience with a standard generalized markup language available to users; with computing systems in their current state of evolution; with the web as the next publishing frontier. My guess is that it would have looked very much like XML does today. For example, take the SGML features of RANK and DATATAG. These were never really used, and the ISO Working Group, had they thought it would have been permissible, probably would have dropped them from the International Standard after the balloting of the Draft back in 1986. Further, today, with cheap memory, fast processors, advances in editorial software, and fairly inexpensive bandwidth, I don't think that the developers would have perceived the need for minimized markup.

SGML was the right standard in 1986

SGML was the right standard in 1986 for its audience. It has stood the test of time and become the foundation for a more streamlined version of itself. Its stability over the first 10 years of its life meant that developers could concentrate on developing most of its features without fearing that the Standard would change. The trade-off was stability against dynamism. It has probably worked out well. If the SGML standard had been a moving object, the level of implementation would have been lower, perhaps making it possible to miss its importance.

4 SGML, XML, and the marketplace

4.1 Industry acceptance

SGML – extensive effect on industry

The effect of SGML on industry has been extensive. When the DOD adopted SGML, they made it financially viable for their suppliers to use SGML not only for their military products, but also for their commercial counterparts. The most obvious area where this progression took place was in the commercial aerospace marketplace.

Airframe manufacturers eventually promoted the use of SGML to the *Air Transport Association (ATA)*, which amended its ATA100 specification (now ATA 2100) to use SGML. After all, if airframe suppliers had to retool to provide SGML deliverables to their military customer, why not also use SGML for their commercial customers, thus providing a greater business case for the conversion to being an 'SGML shop'.

commercial publishers

At the same time that the DOD was moving toward SGML, so were commercial publishers. The *Association of American Publishers (AAP)* published its *Electronic Manuscript Project* in 1986 just before the final Standard was published. Scientific publishers were particularly interested in SGML to help in the areas of tabular and mathematical data.

automotive, semi-conductor, and telecommunications industries

Throughout the late 1980s and early 1990s, more and more industries moved toward SGML including automotive, semiconductor, and telecommunications. The volume of information in SGML has grown steadily. Industry has seen SGML as an industrial-strength solution to an industrial-strength problem.

Just as SGML made significant inroads into various industries, XML has also attracted many industries. Some of these industries had individuals or small numbers of users who had promoted the use of SGML. The visibility of XML, together with its perceived simplicity, has broadened the attractiveness of a generic markup language to more members of an industry and more industries in general. Some examples include:

- financial
- classified ads for newspapers
- speech synthesis
- real estate
- mobile phone interface design
- graphics
- electronic commerce (EDI and retail trade)
- translation memory
- scripting for new broadcasts
- ontology
- astronomy
- genealogical data

4.2 Product marketplace

not an extra-ordinary growth rate of vendors

Companies that embarked on the development of SGML products had a lot of work ahead of them. ISO 8879 was written by publishing professionals, not all of whom were computer scientists. Many of the rules of SGML, while seemingly sensible for a publisher, were counter-intuitive to a computer scientist. The development of parsers, editors, databases, transformation utilities, and browsers was a long, expensive process. Although the numbers of vendors has grown over the years, the rate of growth has not been extraordinary.

The DOD's acceptance of SGML provided the necessary 'carrot' to entice software developers to the market. Yet, even after 12 years, the vendor community has not grown to the extent envisioned. There are still only about 3 or 4 editorial tools, 2 or 3 transformation tools, 2 or 3 databases, and 3 or 4 browsers, widely in use today. SGML tools, despite the wide acceptance by industry, is still considered a 'hard' market to break into.

5 Market impact

SGML is not a sexy standard

So, let's face it, SGML is not considered a 'sexy' standard. It is viewed by those who have not come to love it, as a dry, hard-to-read, idiosyncratic and harder-to-implement standard that appeals to a very niche market. Everyday, people who think SGML is old and boring ride on planes, drive cars, use semi-conductors and computers, and talk on telephones that have been documented in SGML; they look up words in dictionaries and encyclopædias and surf pages on the Web that have all been encoded in SGML. Yet for all its acceptance under the covers, SGML is not something people tend to admit using in public. Why? Probably perception.

SGML – too hard, too confusing, too expensive?

Over the years we've heard: it's too hard; too confusing; too expensive. Yet, as each ardent opponent is converted to an ardent supporter, those who have been promoting and using SGML since the beginning have looked for the magic words needed to make that conversion in perception easier, faster. And what has been found? Nothing ... there are no magic words beyond how using SGML makes good sense for a business because it protects the business' information assets.

XML – preaching exactly the same gospel has become hot

Imagine how much more amazing it then seems, that in less than two years, XML, by preaching exactly the same gospel regarding ownership of data assets and the need to move away from proprietary markup has become, well, 'hot'. XML is SGML, just without the lacy frills. It has been reconstituted without the ability to change the delimiter set and without the 'features'. It has been spruced up to take more notice of accepted computer science practice.

So, what will be XML's market impact? So far, XML has proved to be a marketing star. No matter which angle the media photograph it from, they deem it to look pretty good.

user demand will fuel product development

XML already boasts a growing army of vendors and individuals who are investing in the development of XML tools or the adaptation of SGML tools. Although hard to qualify at this early stage, the indication seems to be that the number of product choices will grow considerably. As in all thriving industries, user demand will fuel product development, and product innovation will excite and attract more users. This cycle repeats and magnifies throughout the growth stage of a market. The XML market is right at the beginning of its lifecycle. I'm sure that we will see an ever-increasing spiral of product innovation and increased use of the XML recommendation over the coming years.

6 Evolution toward more sophisticated information processing

6.1 Component vs document publishing

Many interesting developments have occurred over the past decade or two in publishing. These developments challenge how we think about documents and other information containers. Perhaps the most compelling of these developments is the movement from documents to components.

a new view on documents

In traditional publishing, the document was a single entity. It was complete in and of itself. And, typically, it was opaque. The structures within the document were not explicitly defined in the encoding of the document. SGML, of course, changed all that. Through the use of structure-based and content-based markup, portions of documents could be delimited. Although before SGML it may have

been typical for publishers to keep documents divided into chapters, there was usually not any further decomposition of the information.

SGML helped to change this view of the document from two perspectives. First, there is the inward looking perspective. What was a document, through the use of SGML markup, could be decomposed into several layers. Information that occurred more than once could be aggregated to a single instance that is pointed two multiple times.

components in static or dynamic combinations

Second, from the outward perspective, each component of what used to be a single document, was now available to be part of an unlimited number of other documents. Further, such use of components in myriad combinations could be static or dynamic. A document could exist for years or for moments, depending on the needs of the user.

The capacity of humans to understand this concept is well-illustrated in the World Wide Web. Depending on your perspective, the information on the web might be one, huge, complex, distributed, highly dynamic document, or it is millions of documents, each with an interesting, but non-explicit, set of criteria for determining each document's boundaries.

6.2 Data and metadata

metadata = data about data

Another evolutionary idea is really just an extension of what has been done in the work of computer science relative to data for many years: there's the data and then there's the data about the data, or metadata. In a database, this might mean that we know that a value is 10 and the metadata tells us that it is a date, or a string, or an amount in a given currency.

many levels of metadata exist

In SGML, there has been a tendency to use SGML attributes to capture metadata. So, an element may have attributes to show that it was reviewed by 'John Doe', or that it has a security classification of 'confidential'. However, there are many levels of metadata. One level that has always been a judgement call by implementers is when the 'metadata' is also part of the data. For example, the entire document may be issued on a certain date, or be written by a particular author. These are often elements within the document, but philosophically, they are also information about the document.

metadata takes on primary significance

In the XML world of the web, metadata takes on primary significance as the *raison d'être* for a number of XML applications. There are XML applications that have little or no 'content', but rather are a series of values describing other objects. These include:

- *Desktop Management Task Force (DMTF) CIM (Common Information Model) data* – The proposed model is intended to let mainframes, Windows-based systems, and other disparate systems access and exchange management data via the Web.
- *HTTP Distribution & Replication Protocol* – proposal to improve efficiency & reliability of data distribution over HTTP (will use RDF).
- *Java Speech Markup Language (JSML)* – XML application to annotate text with additional information to improve quality/naturalness of synthesized speech.
- *Open Software Description Format (OSD)* – OSD, an application of the XML, is a vocabulary used for describing software packages and their dependencies for heterogeneous clients.
- *Platform for Privacy Preferences (P3P) Syntax Specification* – P3P applications will enable sites to automatically declare their privacy practices in a way that is understandable to users' browsers.
- *UML (Unified Modeling Language)* – emerging standard modeling language for the description of software systems.
- *Web Interface Definition Language (WIDL)* – specification of an XML application that allows interactions with Web servers to be defined as functional interfaces that can be accessed by remote systems over standard Web protocols and provides structure necessary for generating client code.
- *Translation Memory eXchange (TMX)* – developed under *Localisation Industry Standards Association (LISA)*; allows easier exchange of translation memory data between tools and/or translation vendors.
- *Web Collections using XML* – metadata syntax, an application of XML that is used to describe the properties of some object, such as a WebPage to provide information to an application using the object.
- *The Resource Description Framework (RDF)* – specification governing the interoperability of applications in terms of metadata property sets. It is being developed under the authority of the W3C. The draft specifications from W3C confirm that 'RDF will use XML as the transfer syntax in order to leverage other tools and code bases being built around XML ...'

Whilst SGML focused primarily on documents, many XML proponents unfamiliar with SGML, think of XML primarily for metadata, with little or no concern for its use within the text of a document.

Certainly, users who exploit both the text-encoding and meta-data-encoding properties of SGML and XML stand to gain the most. It will mean that the processors of the contents of documents can use the same parsing engines as the processors that manipulate the network, the file system, the protocols, and the files themselves.

7 Using XML

7.1 Is there trouble in paradise? – design flaws

less expensive products over time

If XML is 'simpler' than SGML, is there a benefit of that simplicity to be reaped when creating an XML system rather than an SGML system? Some benefit will be reflected in less expensive products over time. Further, integration costs will come down eventually as product interoperability is improved. However, many of the costs incurred when creating an SGML system, such as the analysis of business requirements, the document analysis tasks, or the functional specification tasks, will still be incurred with XML.

XML is not a panacea

XML is not a panacea. Design choices, made for very sound reasons during the development of XML, may have consequences during system development. Thus, there will be problems with implementations just as there are with any standard.

problems with Unicode?

An area that might cause some problems in the near term is the use of *Unicode*. The citing of the Unicode standard is an important foundation of XML and, in the long run, will make sure that there is strong, global, character set support. However, in the near term, there is less than comprehensive support for Unicode. Add to that, the few thousand code points that are unassigned, and there may be a few growing pains with the use of Unicode.

Conversely, could the simplicity of XML mean that companies will have to compromise or give up benefits of SGML when using XML? Companies that want to use XML outside the Web may find that the use of a single concrete syntax, the lack of support for *Formal Public Identifiers*, and the requirement for the explicit specification of *System Identifiers* proves limiting compared with SGML.

7.2 XML: SGML for consenting parties

A concept that will probably cause some confusion for XML users is the optional use of DTDs, or perhaps even more confusing, the use of partial DTDs. (I must confess, the wonderful title of this section was suggested by Jeanne El Andaloussi and François Chahuneau of *AIS* (*Advanced Information Systems S.A.*).

a DTD turns the document into a self-describing text base

Until XML came on the scene, SGML could point to the DTD as the primary construct that set it apart from all other generic tagging schemes or macro languages. Because of the DTD, the document was a self-describing text base. Like a database, you could define what could occur in a document class and then check to make sure it did occur in a particular instance by parsing the entire document, including its DTD. For many 'industrial-strength' users, this was the part of SGML that made it all worthwhile. For example, if you are building a fighter jet and have a few hundred authors working on creating the operating and maintenance documentation, the DTD could be used to make sure that they were all using the same structures in the same order and frequency. An information owner could determine the degree of control used in authoring that information by deciding to use either a stricter or more benevolent DTD.

SGML – the quiet revolution

Due to the requirement to have a DTD, SGML created a 'quiet revolution'. (This term was coined by the late Yuri Rubinsky, founder and President of *SoftQuad, Inc*). To use SGML, it was necessary for an organization to think about its documents in ways that it had never done before. The exercise of developing a DTD contributed to a change in the way organizations thought about their documentary information. Prior to SGML, there was documentation planning, but there was little rigor and no way to enforce what little there was. SGML and its use of the DTD changed all that.

XML – optional DTD

When XML was formally announced in Boston at SGML '96 in November of that year, the XML Working Group explained the reasoning behind their decision to make the DTD optional. It was clear to the audience, that perhaps even the members of that Working Group did not totally agree with one another. For example, some members felt that a DTD would, 'of course', still be needed for authoring, but that for delivery, where it is assumed that the instance is already known to be valid, it would not be required. Others felt that being able to author a well-formed document, without a DTD, would be a legitimate requirement and a reasonable way of working.

additional user responsibility

What is clear, is that the minute the XML Working Group made the DTD optional, they made it necessary for XML users to have to bear an additional responsibility. Users must decide if a DTD will be

used in an application, how much of a DTD will be used, and at which points in the production life cycle it will be used. For the individual user or the traditional Web user, the need for this decision may not be obvious. They may decide not to use a DTD, full-stop. However, corporate and commercial publishers will need to analyze their requirements and make informed decisions. In some ways, it's a lot like deregulating an industry. With regulation, life may not be easy, but at least you know where you stand. The rules are clear, and the decisions left to be made are few. With the advent of deregulation, there may be more chaos and more accidents initially, the responsibility is more in the hands of the user, and given time, the users grow into their new role.

weaknesses of the SGML DTD

Oddly enough, the vehicle that brought about the optionality of the DTD may also wind up being the catalyst for the development of a more robust replacement for it, as well. The current SGML DTD is a bit of an anomaly:

1. It plays two roles. It models the structures that can occur in a document instance and it defines the syntax of the markup in the instance, e.g., if there is minimization, etc.
2. It is in a different syntax than an SGML instance.
3. It is fairly limited in scope. For many years now, there have been calls from the user community to extend the modeling capabilities of a DTD, e.g., lexical recognition in attributes values, etc.

fulfilling the roles of a Markup, Modelling, and Management Language

The XML community has put forward a number of alternate suggestions: XML-Data, XSchema (XSD), etc. These proposed DTD replacements tend to focus on the creation of a schema language that uses the syntax for XML instance markup and extends the role of the DTD to provide for inheritance and data typing (see Chahuneau 1998). As a user of SGML, I have noted that SGML moved from 'Standard Generalized *Markup* Language' to 'Standard Generalized *Modelling* Language' to 'Standard Generalized *Management* Language' in the way it was used within industry. However, it fell short of really fulfilling these roles, perhaps due to the inherent weaknesses described by Chahuneau. If XML or SGML were to have a schema language that could be used to model documentary information within the greater milieu of a corporation's overall data model, the transition to the 'document of tomorrow' might more easily be accomplished.

Tomorrow, tomorrow

So, what does the 'document of tomorrow' mean? Over the past two decades, and particularly in the past five years, the corporate and commercial publishing industry has seen a shift in the definition of the document. This definition in some cases may have been pioneered in software using proprietary markup schemes. However, it is

only through the use of well-structured documents, using generic markup languages, that it will be possible to realize a world-wide use of these new documents.

8 Documents as dynamic knowledge bases

knowledge = value-added information

Following on from the earlier discussion of component-based publishing, documents will continue to evolve as 'dynamic knowledge bases'. The term knowledge here refers to 'value added' information. Building on the earlier example, 10 is a datum. If we know that 10 is the value in a given currency, we have information. If we can link that value to the value last year, the value in another currency, and an indication of the market's perception of that value, we have added value to that information and now have knowledge about it. We understand it in a greater context. We can base judgements on it, we can make informed decisions.

document model simulating how we think

Today, we still struggle to find out all we need to know to interpret information. The use of structured documents; generic, content-based markup; extensive and implied hyperlinks; accessible schemas; and highly networked or interconnected systems will increase our ability to acquire knowledge. The concept of the 'all-in-one', linear document will loose its grip on our loyalty; we will move perhaps naturally toward this definition of a document over time. Why naturally, because this model more closely simulates how we think.

In his groundbreaking article, *As We May Think*, Vannevar Bush (Bush 1945) described his idea for a way to access information consistent with how our minds naturally work. He writes:

> The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain.

He continues:

> Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and to coin one at random, *memex* will do. A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.

> Wholly new forms of encyclopædias will appear, ready-made with a mesh of associative trails running through them, ready to be dropped into the memex and there amplified.

His conclusion, therefore:

> Thus science may implement the ways in which man produces, stores, and consults the record of the race.

He even speculates on how this will affect the human race:

> Presumably man's spirit should be elevated if he can better review his shady past and analyze more completely and objectively his present problems. He has built a civilization so complex that he needs to mechanize his record more fully if he is to push his experiment to its logical conclusion and not merely become bogged down part way there by overtaxing his limited memory. His excursion may be more enjoyable if he can reacquire the privilege of forgetting the manifold things he does not need to have immediately at hand, with some assurance that he can find them again if they prove important.

close to create Vannevar Bush's Memex

Now, 50 years later, we are at a stage of technological evolution where it is possible to create a 'memex', in large part because of the use of structured documents and generic markup languages.

8.1 Information (knowledge) engineers

authors as information engineers

Creating webs of information that provide 'knowledge' to the user of those webs will mean that the field of authoring must undergo a metamorphosis. Authors will be more accurately identified as information engineers. Whilst the original development of text will still be an important activity, the ability to create, enhance and maintain appropriate links to associated information will be critical to the realization of dynamic knowledge bases. We will see 'authoring' tools change dramatically as they provide for this second function of the author.

Behind the scenes, SGML and XML will be making much of this information engineering possible. They will provide the consistent information encoding standards. Their associated standards, such as HyTime, DSSSL, XLink, XSL, schema, and metadata standards, will make it possible to define global links and to present information in comprehensible formats.

9 Conclusions

SGML and XML are promoting the same basic principles

For those who have been pushing for SGML to become widely accepted for the past decade, XML's popularity should be good news. XML promotes the same basic principles of SGML (limited dependence on proprietary systems, investment in a business' data assets, application of structure to text objects, etc.). It's perceived 'ease of use' should attract more users. As a company's understanding of how a generic markup language can help them to meet their business goals grows, their demands on that language may escalate. The users' increasing sophistication will lead one of two places: a move to the use of the full SGML standard or a demand for an enhancement of the XML standard.

keeping track of the differences will be too cumbersome

My personal prediction is that for the majority of industrial users, keeping track of the differences between the two languages will be too bothersome. People will use the term XML to mean a standard, generic markup language with all the characteristics that such a phrase implies. For some, they will use an XML layer on an SGML product and will at times use something that is only strictly available in SGML. Will they know that? No. Will they care? No. The point will be that the use of a generic markup language is helping them to structure their documentary information and thus returning business benefit.

companies with SGML capability will enjoy the greatest flexibility

Those companies that currently have an SGML capability will enjoy the greatest flexibility in the future: to retain information in SGML, to transition source data to XML, to only use XML as an output format, to transform to HTML, to deliver using proprietary products. All of these are legitimate options. The investment in SGML has been a sound strategic choice and it continues to be because of this very flexibility. Companies just starting to use structured documents now have a choice of either SGML or XML, or both, depending on their business requirements.

the Web can return significant business benefits

The use of delivery platforms such as the Web can return significant business benefits not because they mimic a paper manual, but because they can be exploited to turn a set of information into a knowledge base. This is the 'added value' gained from providing the user with the ability to quickly navigate across disparate classes of information, to find important relationships and additional detail. For the corporate publisher delivering operational and maintenance information, increased use and dependence on the deployed knowledge base leads to greater consistency of actions across a shifting and perhaps unevenly skilled set of personnel, resulting in fewer safety incidents and less down time. For the commercial publisher, the

ability to provide a more dynamic and rich information set means greater loyalty amongst buyers and subscribers.

Content markup, coupled with the use of small modules of information and well-defined metadata identifiers comprise a powerful arsenal for the creation, management, manipulation, and deployment of corporate knowledge. Taken together, these form the basic foundation for companies to:

- plan and execute the creation of a complete aggregate of corporate information,
- manage that information over its entire life including updates and shifts in its internal relationships, and
- deliver that knowledge base in a variety of ways that ensure use by and feedback from users.

The ultimate promise

The ultimate promise of the widespread use a generic markup language is a business environment where:

- information travels as freely and easily as desired,
- content, structure, and movement of that information can be as controlled as desired, and
- information in a file, the information about the file itself, the processes performed on that information, the results of the processes, and the environment in which it all takes place is a seamless continuum.

Prior to the advent of the World-Wide Web, we did not think in these terms. SGML could not have kept this promise in its original state, primarily due to its idiosyncrasies in computer science terms. XML could not have been created as quickly nor as well, without the experience gained worldwide by the use of SGML. XML, or whatever it evolves into, will see the realization of this promise.

References

(Bush 1945)
Bush, Vannevar: As We May Think. In: The Atlantic Monthly (July 1945)

(CALS)
Originally, in 1985, CALS stood for Computer Aided Logistic Support, an initiative created by the US Department of Defense. Shortly after the term was coined, the meaning was changed to Computer-aided Acquisition and Logistic Support. In the early 90's, the meaning was again modified to Continuous Acquisition and Lifecycle Support and later to Commerce At Lightening Speed. The goal of the CALS initiative was to have all documentation in support of a weapon system available in electronic format. This implied the

use of electronic data standards. By 1990, other NATO members were beginning to examine the use of CALS concepts and these ideas were also being pushed out to the private sector.

(Chahuneau 1998)
Chahuneau, François: SGML schemas: From SGML DTDs to XML-DATA. In: Conference Proceedings SGML/XML Europe '98, 1998

(SGML Users' Group 1989)
SGML Users' Group (Ed.): A Brief History of the Development of SGML., June 1989